20 VIDEOLEZIONI DI BASIC
PER IMPARARE CON LO SPECTRUM

GRUPPO
EDITORIALE
JACKSON

Spectrum

16K/48K/PLUS

VIDEO BASIC SPECTRUM
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea - Via Indipendenza 88 - Como
Redazione software:
Francesco Franceschini, Roberto Rossi,
Alberto Parodi, Luca Valnegri
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di
Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70
(autorizzazione della Direzione Provinciale delle
PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo
Editoriale Jackson S.r.l. - Via Rosellini, 12
20124 Milano, mediante emissione di assegno
bancario o cartolina vaglia oppure
utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere
richiesti direttamente all'editore
inviando L. 10.000 cdu. mediante assegno
bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

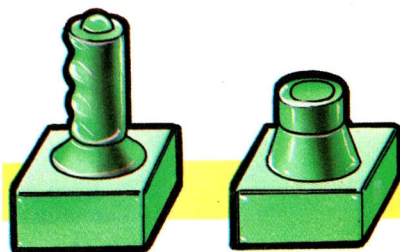
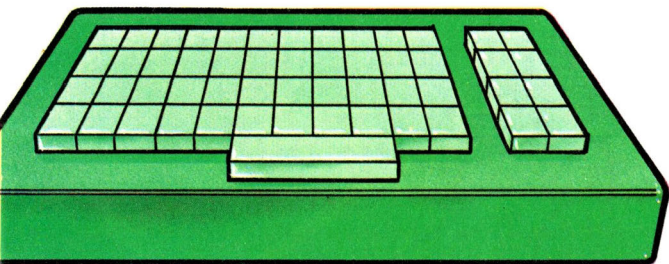
Gruppo Editoriale Jackson

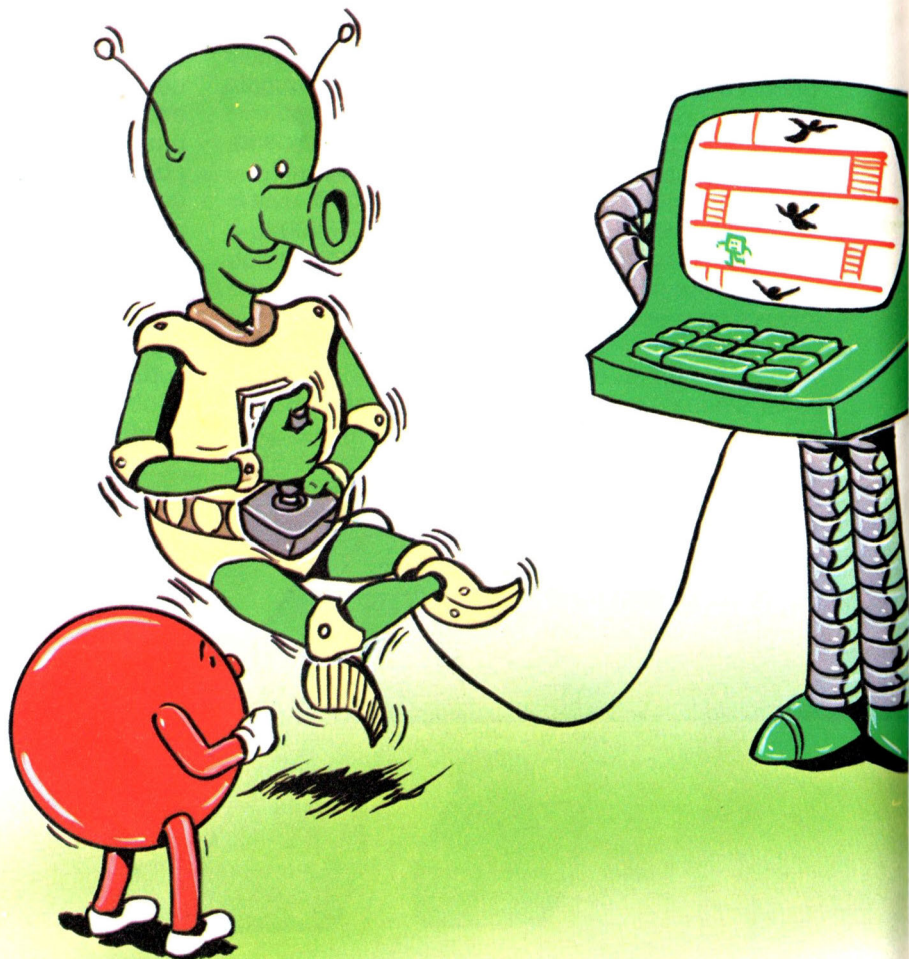
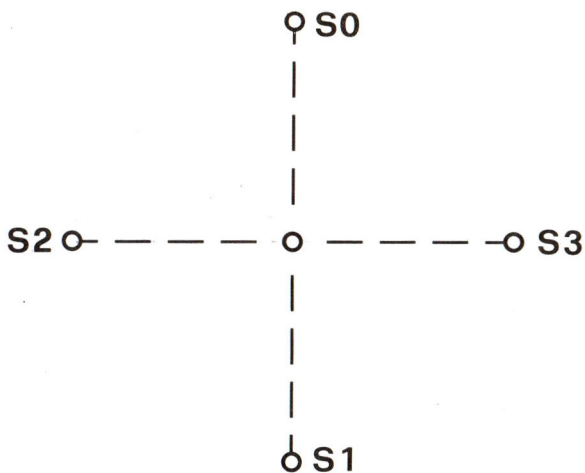
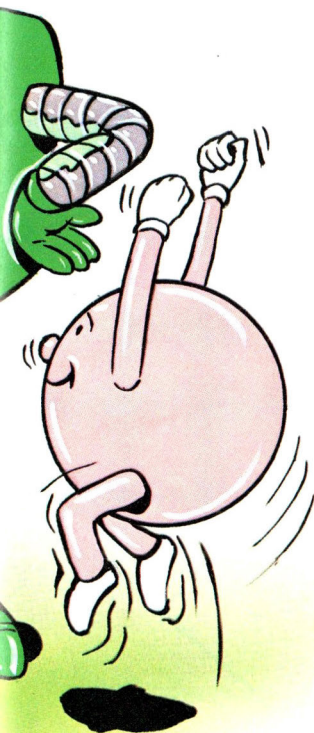
SOMMARIO

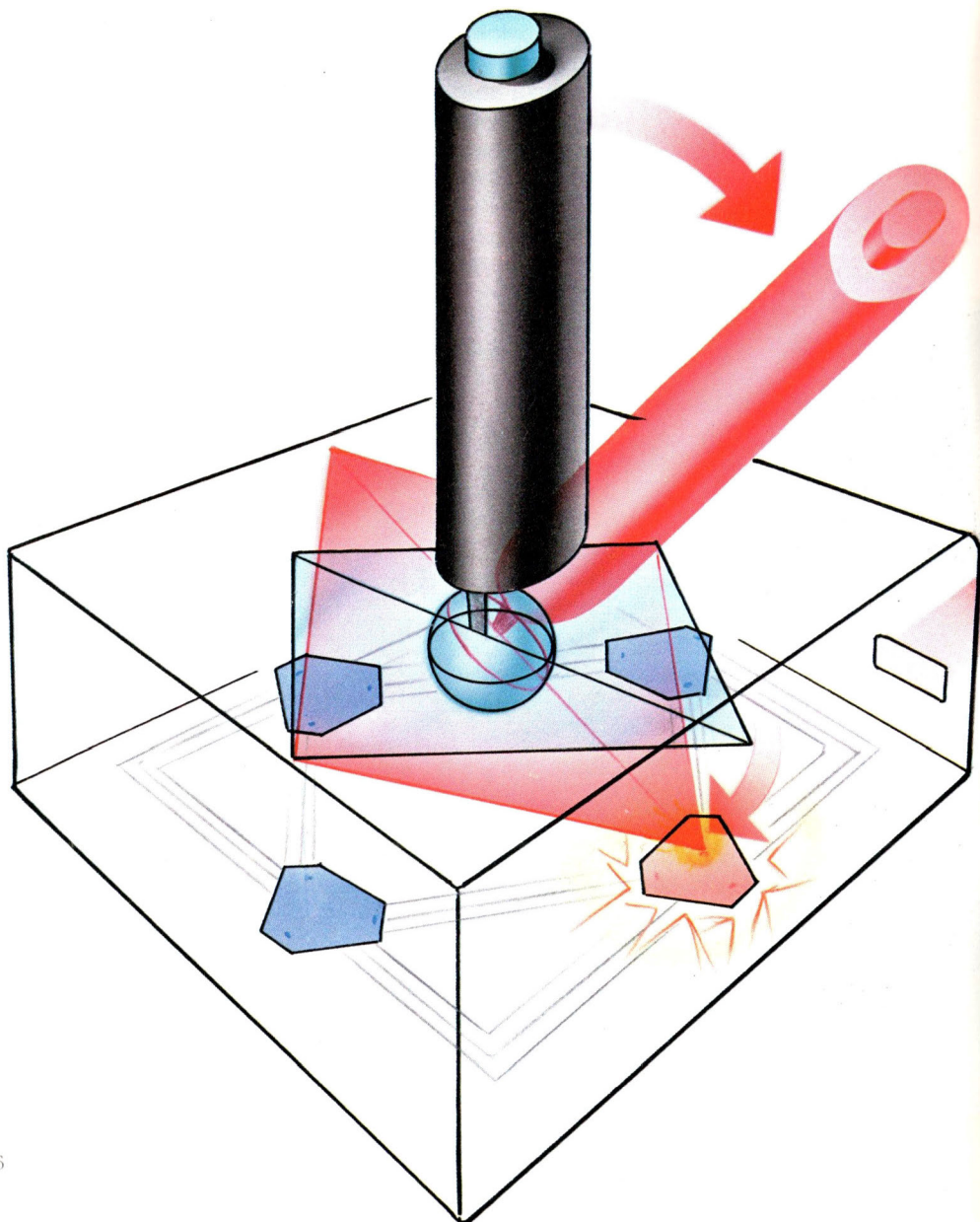
Introduzione

Chi non conosce il joystick, quella incandescente leva con pulsante onnipresente nei videogiochi da casa o da bar. Quante battaglie vinte o perse impugnandolo contro terribili nemici. Al di là del gioco, comunque il joystick (o le sorelle paddle) è una periferica di ingresso, con la sua logica (digitale) e i suoi principi di funzionamento, che è bene conoscere. Questo per quanto riguarda l'hardware.
Per la programmazione un altro prezioso tassello si aggiunge alle conoscenze già fatte: le funzioni di stringa.
Finora abbiamo limitato l'uso delle stringhe all'assegnazione, confronto e stampa. Vedremo in questa lezione come poter manipolare dei dati alfanumerici grazie a LEN, VAL, STR$, TO, IN/OUT.

Paddle e Joystick

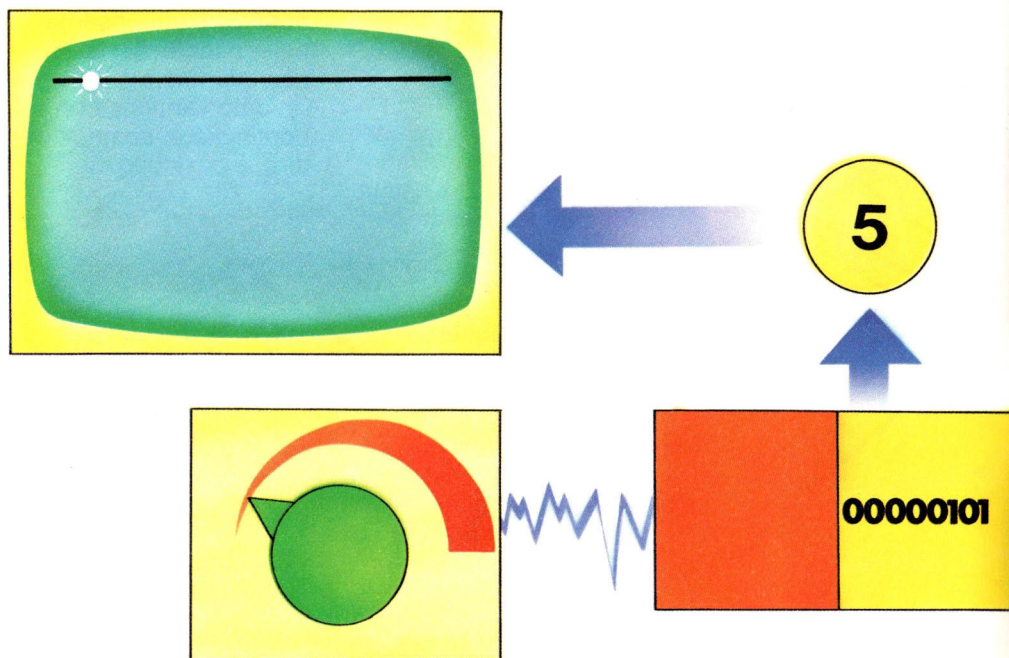
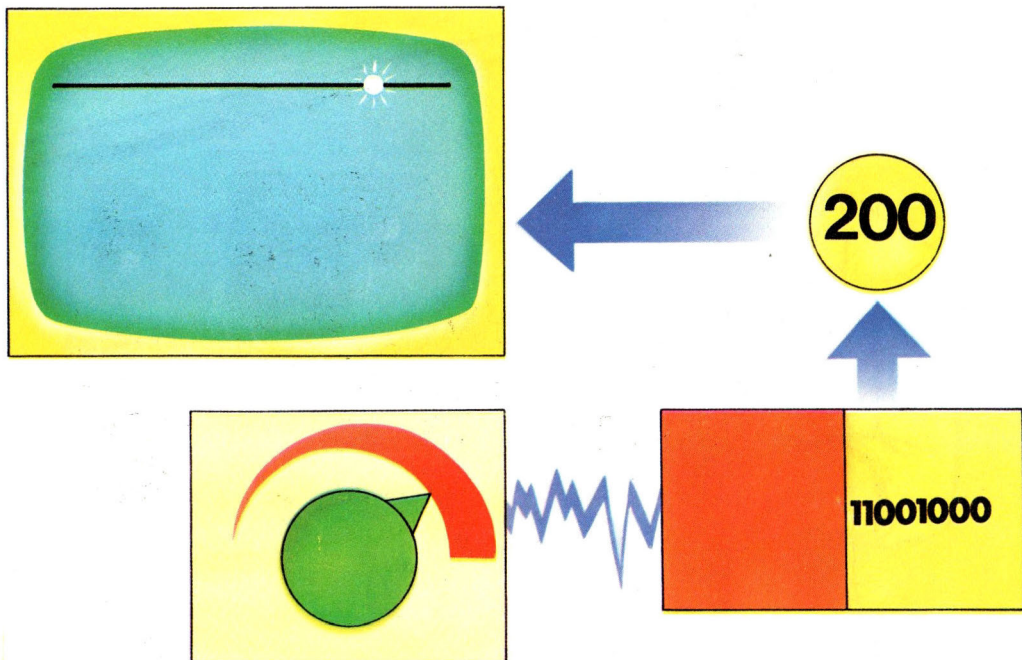
Abbiamo visto nelle scorse lezioni che il principale dispositivo di ingresso di dati ed informazioni per un calcolatore è costituito dalla tastiera, tant'è che praticamente non esistono computer - eccettuati quelli costruiti per particolari applicazioni - che non ne dispongono e che per l'input dei dati debbono ricorrere a mezzi differenti.
Principale non vuole però dire unico: sono infatti disponibili numerosi altri accessori e periferiche che in particolari circostanze possono rendersi molto più utili ed adeguati della tastiera, proprio come accade per l'uomo quando desidera qualcosa di più veloce o comodo della semplice (ma assolutamente indispensabile) penna per scrivere.
Oggi parleremo appunto di due di questi dispositivi: i joystick e le paddle.
Con ogni probabilità essi non ti sono completamente sconosciuti, dal momento che i loro nomi ricorrono molto spesso tra gli accaniti giocatori degli ormai diffusissimi ed onnipresenti videogiochi, essendo infatti utilizzati come principali armi di attacco e di difesa contro i vari invasori spaziali.
Anche chiunque sia entrato in un qualsiasi bar non può non conoscere un joystick, riconoscendo subito in tale nome l'incandescente levetta necessaria per evitare sfuggire dai pericoli del "bombardamento

nemico".
Più facilmente non ti sarà invece molto noto il loro principio costruttivo e di funzionamento: l'obiettivo di questa lezione sarà appunto quello di spiegartene i segreti e le differenze.
Cerchiamo innanzitutto di vedere quali sono stati i motivi che hanno determinato la nascita e, soprattutto negli ultimi tempi, la diffusione dei joystick e delle paddle.
In pratica, tutto è cominciato pochi anni fa, proprio con i fatidici videogiochi: un videogioco, infatti, altro non è che un particolare calcolatore programmato per eseguire solo ed esclusivamente un certo programma; cioè un calcolatore, come si usa dire, "dedicato". Il suo programma è appunto il gioco.
Non ci volle molto per rendersi conto che era necessario superare la difficoltà della tastiera: in primo luogo la maggior parte dei tasti sarebbe stata assolutamente superflua per l'uso al quale era destinato il calcolatore ed inoltre avrebbe reso molto più semplice, e soprattutto divertente, per gli utilizzatori poter avere l'impressione di "governare" direttamente, proprio come accade sulle astronavi vere, attraverso una specie di leva di comando ed un pulsante di fuoco. Attraverso questi accessori divenne pertanto possibile impartire ordini alla macchina, in modo rapido, facile e soprattutto senza conoscere nulla di programmazione.

Come controllare un joystick

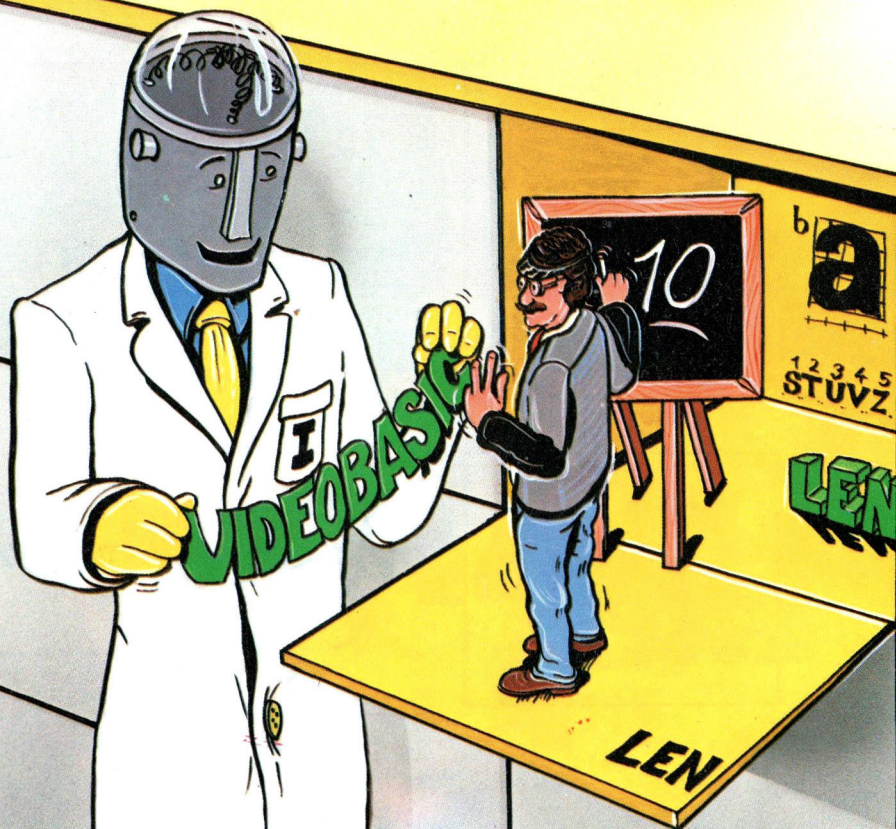
Per una persona non è molto naturale premere dei pulsanti per comandare dei movimenti: non si avrebbe la continua percezione di movimento che ricaviamo per esempio ruotando il volante di un'automobile se dovessimo invece utilizzare dei tasti o dei pulsanti.
Fu così che nacque l'idea dei joystick e, conseguentemente, delle paddle (che vedremo essere una specie di joystick piu raffinati).
Attualmente l'uso di questi dispositivi non è più prerogativa esclusiva

dei videogiochi: quasi tutti i personal computer ne permettono difatti la connessione, attraverso alcuni collegamenti appositamente realizzati sul calcolatore stesso. Un joystick è pertanto - alla stessa stregua della tastiera - un dispositivo di input; esso appare come una leva che con il suo movimento può permettere di spostare un oggetto sul video. Vediamo in quale modo. All'interno del joystick esistono quattro interruttori, che vengono azionati in dipendenza della direzione assunta dalla leva; in base al loro stato è perciò possibile identificare nove diverse posizioni della leva:

tutti aperti	leva verticale
S0 chiuso	nord
S1 chiuso	sud
S2 chiuso	ovest
S3 chiuso	est
S0, S1 chiusi	nord-ovest
S1, S2 chiusi	sud-ovest
S2, S3 chiusi	sud-est
S3, S0 chiusi	nord-est

A ciascuno di questi quattro interruttori viene associato, in una locazione della memoria all'interno del calcolatore, un bit: bit 1 per interruttore acceso o bit 0 per interruttore spento. Un quinto interruttore funge da

pulsante di fuoco o da segnalatore che la posizione desiderata è stata raggiunta.
Tramite programma basterà allora leggere nella locazione il valore di ciascuno di questi bit per risalire alla posizione della leva e quindi alla relativa azione da eseguire (spostamento del cursore, dell'astronave o dell'oggetto sul video).

Informazioni analogiche e digitali: interfaccia

Il joystick è pertanto un tipico dispositivo digitale: il dato contenuto nella celletta di memoria è infatti l'esatto corrispondente della combinazione di interruttori conseguente all'azionamento della leva. Lo stato di ciascun bit dipende cioè dallo stato del rispettivo interruttore, senza nessuna approssimazione od arrotondamento.
Talvolta risulta però più comodo poter disporre di un dispositivo più sensibile a piccole variazioni e spostamenti, proprio come accade con un volante di automobile. Al joystick propriamente detto è stato quindi affiancato il cosiddetto joystick a potenziometro (paddle).
Ai fini del funzionamento una paddle è molto simile ad un joystick: tuttavia, anziché esservi una leva, nella paddle compare una manopola, la cui regolazione risulta pertanto molto più fine e precisa. La cosa viene inoltre ottenuta in modo completamente diverso da come accade con il joystick: lo spostamento della manopola provoca infatti non più l'azionamento di interruttori e pulsanti, ma delle variazioni di tensione ai capi di due resistenze variabili (o potenziometri) inserite nella paddle stessa. Il risultato finale è pertanto un dato variabile con continuità o, come si usa dire, è costituito da

un'informazione analogica. Un'opportuna conversione di tale informazione da analogica in digitale fornisce quindi, sotto forma di numero a otto bit, l'azione specificata dallo spostamento della manopola, memorizzandola in una particolare locazione della memoria. Per eseguire questa conversione viene naturalmente utilizzata un'apposita interfaccia, posta tra unità centrale e paddle, che ha lo scopo di trasformare il segnale continuo fornito dalla paddle in un'informazione binaria. Attraverso un'operazione di lettura, in questa locazione sarà allora possibile, analogamente a quanto succede col joystick, risalire al movimento desiderato.

Naturalmente, anche una paddle ha un suo limite di sensibilità, dovuto proprio a tale operazione di conversione: esso risulta comunque estremamente al di sopra di quello del joystick, permettendo in totale 256 possibili combinazioni. Non è però assolutamente detto che una paddle sia meglio di un joystick; dipende infatti dall'uso e dalla sensibilità necessaria alla singola applicazione. Ciò che sicuramente è più semplice è la connessione del joystick al computer, poiché non richiede alcun dispositivo di conversione, come invece è il caso della paddle.

Gli operatori di stringa

Fino a questo momento il nostro utilizzo delle stringhe si è limitato soltanto all'assegnazione, al confronto ed alla stampa delle espressioni di questo tipo. In molti programmi per computer compaiono invece, in maniera estensiva, manipolazioni e trattamenti dei dati alfanumerici. In questa importante lezione cercheremo quindi di imparare le istruzioni e i comandi che consentono di effettuare simili operazioni.

LEN

La funzione LEN (abbreviazione dell'inglese LENght, lunghezza) è molto semplice da comprendere: consente infatti di trovare il numero di caratteri contenuti in una stringa.
Per esempio, volendo determinare il numero di caratteri che compongono la stringa "GIOVANNI", sarà sufficiente impartire:

```
PRINT LEN "GIOVANNI"
```

e sullo schermo apparirà 8.
La stringa "GIOVANNI", della quale vogliamo contare i caratteri, è l'argomento di LEN e va quindi messa entro parentesi, subito dopo la parola LEN.
LEN restituisce quindi un risultato numerico, partendo da un argomento rigorosamente di tipo stringa, variabile o costante. Da notare che le virgolette non vengono considerate (e d'altra parte è anche giusto e logico) come costituenti la stringa.
È un comando molto potente e di utilizzo assai frequente: i possibili casi nei quali può interessare conoscere la lunghezza di una stringa sono infatti numerosissimi, tra i quali - per esempio - gli incolonnamenti, le impaginazioni o i confronti tra stringhe.

Esempi

```
PRINT LEN "CONTO CORRENTE"
```

Stampa il numero 14.

```
LET A = LEN ("")
```

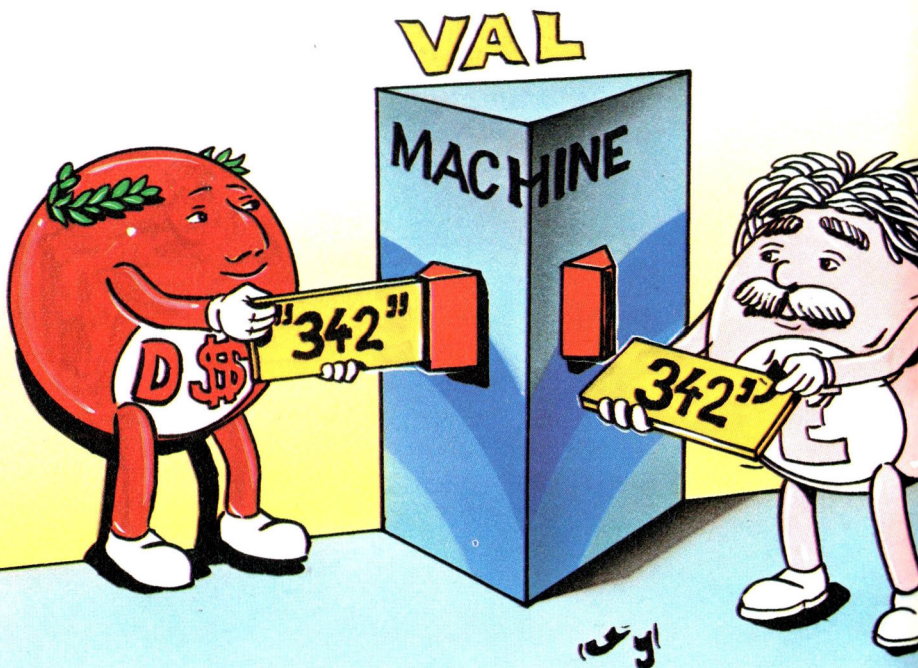
Alla variabile viene assegnato valore 0 (la stringa nulla non contiene alcun carattere).

```
LET A$ = "GIAN"
PRINT LEN (A$ + "CARLO")
```

```
LET A$ = "GIAN"
PRINT LEN (A$) + LEN ("CARLO")
```

Queste due istruzioni hanno lo stesso risultato, cioè 9. Nel primo caso viene prima eseguita la concatenazione delle due stringhe e quindi il calcolo della lunghezza, mentre nel secondo vengono separatamente calcolate le due lunghezze e quindi sommate.

Sintassi della funzione

LEN (stringa)

VAL

Vi sono alcuni casi in cui risulta molto comodo poter convertire una stringa di numeri (come "3149") in un valore numerico, così da poterne far uso in espressioni numeriche. La funzione VAL (abbreviazione di VALue, valore) converte le stringhe in numeri. Così:

```
LET A = VAL "3149"
```

assegna alla variabile numerica A il valore 3149.
Non bisogna comunque fraintendere questa funzione: essa non esegue infatti alcuna magia. Una stringa come "TAPPO" non potrà pertanto mai assumere alcun valore numerico; quando l'interprete BASIC incontra un'istruzione come

```
PRINT VAL "TAPPO"
```

va a controllare se esiste una variabile con quel nome.
Nel caso non la trovi manda il messaggio:

VARIABLE NOT ROUND

Se però la variabile è stata precedentemente assegnata ne stampa il valore.

Esempi

```
LET X = VAL "8"
```

Assegna alla variabile X il valore *numerico* 8.

```
PRINT VAL "54C3"
```

Provocherà il messaggio di errore NONSENSE IN BASIC in quanto la stringa contiene un carattere non numerico.

```
LET P$ = "103"
LET Q$ = "2"
PRINT P$ + Q$
```

Stampa la somma (cioè la concatenazione) delle stringhe P$ e Q$. Quindi il risultato è la stringa "1032".

```
PRINT VAL P$ + VAL Q$
```

Stampa la somma dei valori numerici 103 e 2, cioè 105.

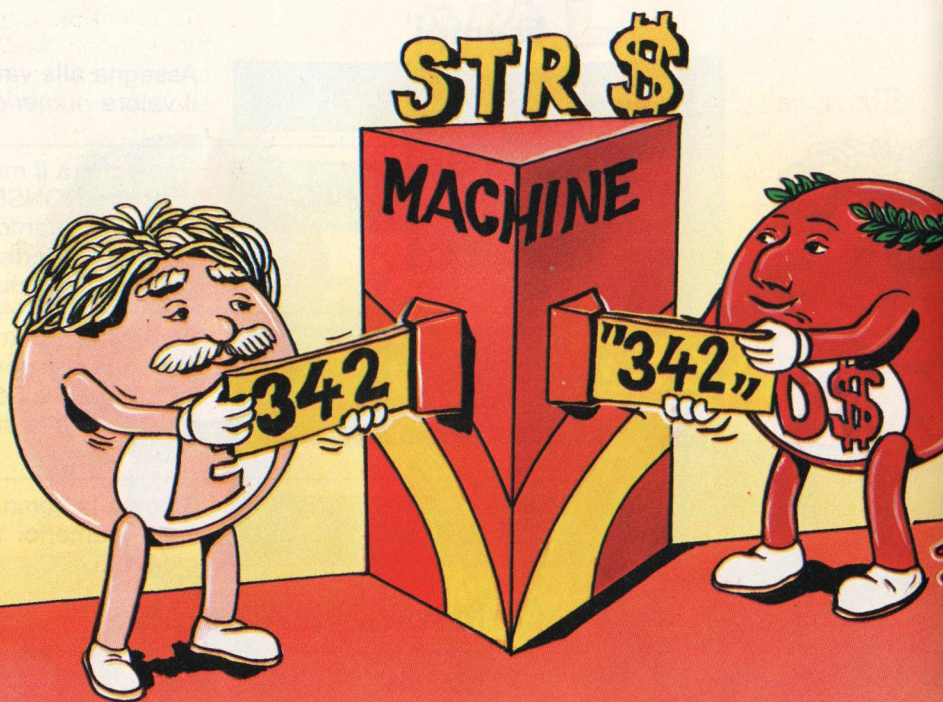
Sintassi della funzione

VAL (stringa)

STR$

STR$ (abbreviazione di STRing, stringa) è la funzione reciproca di VAL: essa consente infatti di convertire una quantità numerica in una stringa di caratteri.
Quando desideriamo trasformare un numero (come 8) in una stringa, in modo che possa essere combinato con altre stringhe (per esempio "mila"), STR$ è la funzione che fa al caso nostro.
L'istruzione

```
PRINT STR$ 8 + " " + "MILA"
```

produrrà allora come risultato la stampa della stringa "8 MILA".

Esempi

LET A$ = STR$ 50	Assegna ad A$ il valore alfanumerico "50".
PRINT STR$ (3 * 7 – 2)	Risultato: 19.
PRINT VAL STR$ 50	È un'istruzione inutile, per quanto perfettamente lecita. Il valore numerico 50 viene infatti prima convertito in stringa (con STR$) e quindi ritrasformato in numero. Sullo schermo comparirà allora 50, inteso come numero, non come stringa.

Sintassi della funzione

STR$ (espressione)

TO

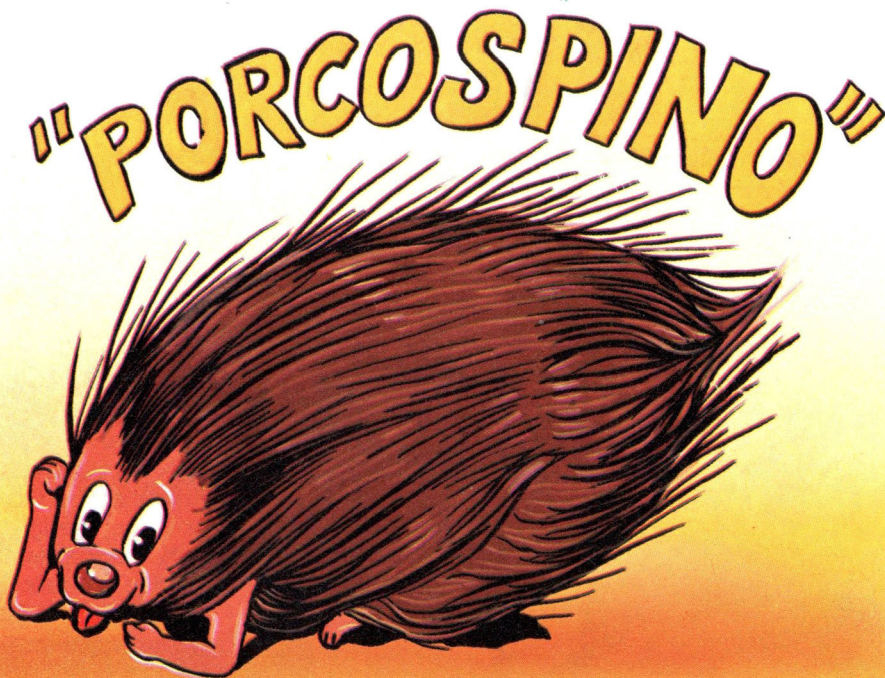
L'operazione di concatenamento permette di aggiungere caratteri ad una stringa (ricordandosi comunque di non superare il limite massimo di 255 caratteri!) eseguendo cioè una sorta di somma tra stringhe.
In certi casi, anziché aggiungere, potrebbe invece essere necessario dover estrarre dei caratteri da una stringa: esiste allora nel linguaggio BASIC del tuo Spectrum una funzione mediante la quale è possibile eseguire questa operazione di separazione: TO.
Il risultato di TO è una sottostringa (cioè un gruppo di caratteri appartenenti alla stringa originale) estratta a partire da una certa posizione specificata e lunga tanti caratteri quanti ne compaiono fino alla posizione dove si vuole che la sottostringa termini.
Vediamo subito un esempio chiarificatore:

```
PRINT "GIAN CARLO" (1 TO 4)
```

Stampa quattro caratteri della stringa "GIAN

CARLO", partendo dalla prima posizione e terminando alla quarta. Il risultato sarà quindi la visualizzazione di GIAN. Il primo numero entro parentesi specifica la posizione del carattere da cui cominciare l'estrazione della sottostringa. Il secondo numero identifica invece la posizione del carattere in corrispondenza del quale l'estrazione deve essere arrestata.
Se ora volessimo eseguire la stessa operazione, ma dalla parte opposta della stringa (cioè se volessimo separare gli ultimi cinque caratteri), basterebbe fare:

```
PRINT "GIAN CARLO" (6 TO 10)
```

ed il risultato sarebbe la comparsa sul video di CARLO.
In genere lo slicing di una stringa, cioè la suddivisione di una stringa in una o più sottostringhe, obbedisce alla seguente forma:
Stringa (posizione primocarattere TO posizione ultimo carattere)

dove i numeri delle posizioni del primo e dell'ultimo carattere specificano dove iniziare e terminare la suddivisione.

Ad ogni modo, visto il gran numero di volte in cui l'operazione di slicing interessa solo un certo numero di caratteri all'inizio od al termine di una stringa, la Sinclair ha reso possibile utilizzare delle forme abbreviate per semplificare la scrittura di questo comando. Così:

```
LET A$ = "RADIOSVEGLIA" (1 TO 5)
```

è equivalente a

```
LET A$ = "RADIOSVEGLIA" (TO 5)
```

In entrambi i casi la suddivisione comincerà dal primo carattere e terminerà al quinto, assegnando così alla variabile A$ la sottostringa "RADIO".
Per assegnare invece ad A$ la sottostringa "SVEGLIA" avremmo potuto scrivere:

```
LET A$ = "RADIOSVEGLIA" (6 TO 12)
```

oppure

```
LET A$ = "RADIOSVEGLIA" (6 TO)
```

Il nostro Spectrum avrebbe sempre capito la medesima cosa: estrarre gli ultimi 7 caratteri dalla stringa di partenza.
Se adesso volessimo ottenere come sottostringa un solo carattere (per esempio la L di MILANO), dovremmo scrivere:

```
PRINT "MILANO" (3 TO 3)
```

od anche

```
PRINT "MILANO" (3)
```

ed il terzo carattere di MILANO, cioè la L, sarebbe visualizzata sullo schermo.

Devi comunque sempre valutare molto attentamente i numeri che segnalano la posizione dei caratteri per l'inizio ed il termine dell'estrazione: valori errati od indesiderati possono infatti portare a suddivisioni assurde o, peggio ancora, a messaggi di errore. Per esempio:

```
LET F$ = "GRUPPO EDITORIALE JACKSON" (19 TO 30)
```

è un'istruzione errata, dal momento che la stringa contiene in tutto 25 caratteri compresi i due caratteri spazio. Anche

```
LET F$ = "GRUPPO EDITORIALE JACKSON" (− 1 TO 6)
```

è errata: nessuna parola può contenere caratteri alla posizione − 1! Per quanto possano sembrarti molto semplici da evitare, questi tipi di errore sono invece di estrema frequenza: spesse volte all'interno delle parentesi non compare infatti un numero vero e proprio, ma una espressione numerica che - passando da un'esecuzione all'altra - può assumere valori diversi. Se non saranno state prese in

considerazione tutte le possibili eventualità, potrà allora accadere di incappare in uno di questi casi, provocando l'insorgere del fatidico messaggio di errore.

INTEGER OUT OF RANGE

L'utilizzo classico di questo comando lo si ha in quei programmi che richiedono in ingresso una data scritta nella forma GG/MM/AA (come 27/10/60). Con TO è allora molto facile scomporre la data nelle sue tre componenti GG, MM e AA.

"PORCO SPINO"

Vediamo subito come fare:

```
10 INPUT "SCRIVI LA DATA (GG/MM/AA)"; D$
20 G$ = D$ (1 TO 2)
30 M$ = D$ (4 TO 5)
40 A$ = D$ (7 TO 8)
50 PRINT G$; " "; M$; " "; A$
```

Alla fine di questo programma G$, M$ e A$ conterranno rispettivamente il giorno, il mese e l'anno della data inizialmente assegnata a D$.

Esempi

```
LET A$ = "gennaiofebbraiomarzo"
PRINT A$ (1 TO 7)
PRINT A$ (TO 7)
```

Stampa "gennaio"

```
LET A$ = "gennaiofebbraiomarzo"
PRINT A$ (16 TO 20)
PRINT A$ (16 TO)
```

Stampa "marzo"

```
LET A$ = "gennaiofebbraiomarzo"
PRINT A$ (8 TO 15)
```

Stampa "febbraio"

```
PRINT "NASTRO" (TO LEN ("NASTRO") – 1)
```

Il numero di caratteri da prelevare partendo da sinistra sarà 5 (infatti LEN ("NASTRO") dà 6, che, diminuito di 1, risulta poi 5).
Sullo schermo apparirà allora "NASTR".

```
PRINT "GIORNO" (1 TO 6)
```

Poiché il valore 6 è uguale alla lunghezza totale della stringa, verrà stampata la stringa completa, cioè "GIORNO".

```
LET A$ = A$ (– 1)
```

Errore! Non è possibile estrarre partendo dal carattere – 1 di una stringa.

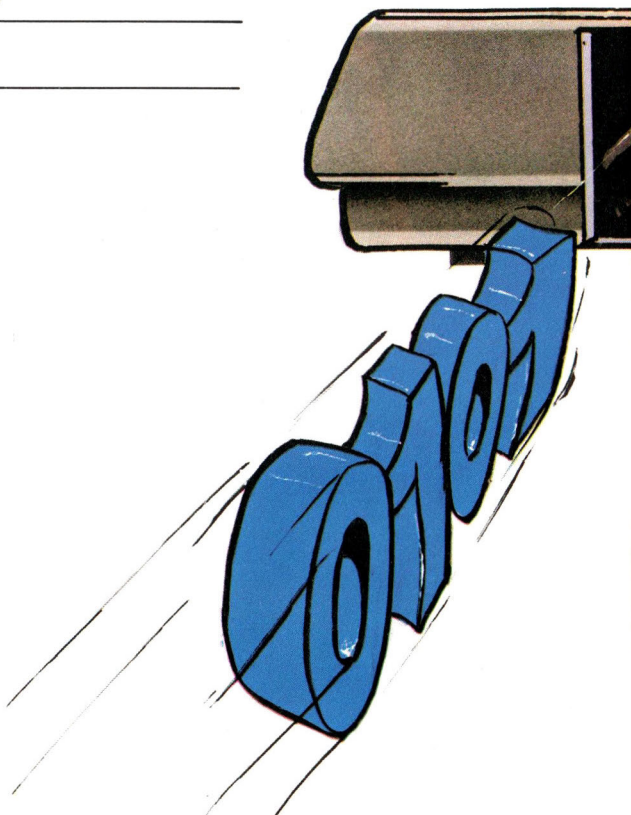
Il seguente, breve programma (altri ne troverai sulla parte della lezione dedicata alla programmazione) ti illustra una semplice applicazione dei concetti finora esposti:

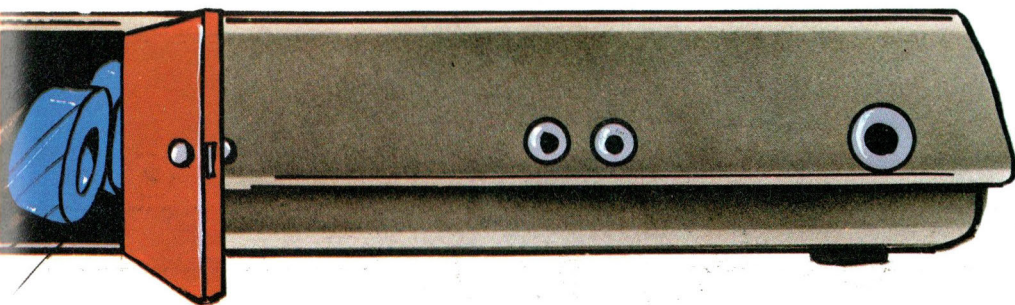
```
10 INPUT "SCRIVI UNA PAROLA QUALSIASI"; P$
20 FOR I = 1 TO LEN (P$)
30 PRINT P$ (1 TO I); " "; P$ (LEN (P$) – I + 1 TO
   LEN (P$))
40 NEXT I
50 STOP
```

Sintassi della funzione

stringa (numero1 TO numero2)

IN/OUT

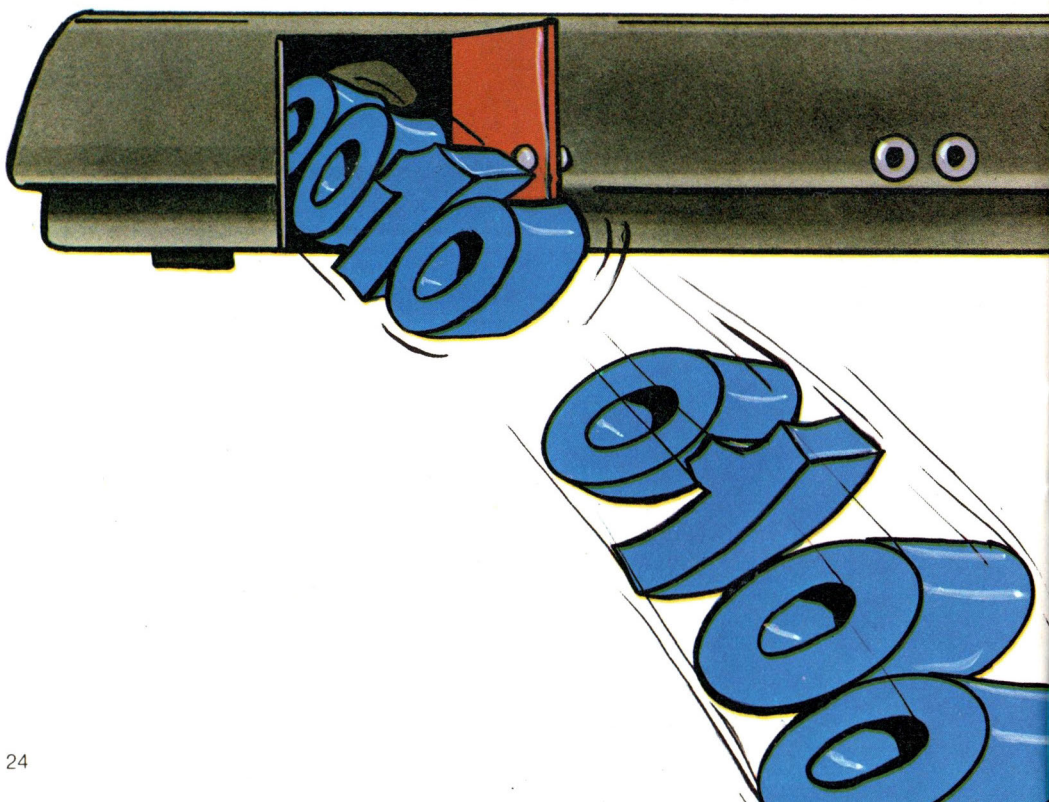
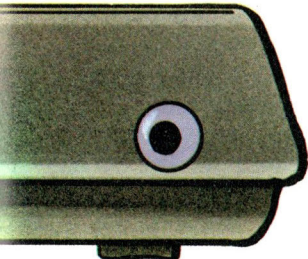
Hai visto che il microprocessore all'interno del tuo Spectrum è in grado di collegarsi - attraverso particolari codici e combinazioni - con qualsiasi parte della memoria: può cioè

accedere (leggendo ed eventualmente scrivendo) a tutte le singole locazioni che appunto compongono la memoria.

Il BASIC ti permette allora - proprio sfruttando questa capacità - di "curiosare" nelle varie locazioni con le due istruzioni PEEK e POKE.
In modo completamente analogo è possibile considerare le varie cellette della memoria non più come semplici magazzini di informazioni, bensì come delle vere e proprie porte di comunicazione tra il computer ed il mondo esterno (che può per esempio essere, a seconda dei casi, la tastiera, la stampante, un joystick od il registratore).
Lo Spectrum fornisce pertanto due comandi per accedere ai singoli dispositivi di input/output, proprio come consente - attraverso PEEK e POKE - di riferirsi a particolari indirizzi della memoria.

IN è la funzione corrispondente a PEEK:

```
IN indirizzo
```

ha infatti un solo argomento (l'indirizzo della porta) e fornisce il valore numerico letto dal dispositivo connesso a quella porta.
OUT è invece il comando corrispondente a POKE:

```
OUT indirizzo, valore
```

scrive il valore (numerico) nella porta specificata dall'indirizzo.
La principale differenza tra PEEK/POKE e IN/OUT è che, mentre tutte le locazioni della memoria si comportano più o meno nella stessa maniera (alcune permettono solo di essere lette, altre di essere sia lette che scritte), il dispositivo che si trova ad "indirizzo" può agire, a seconda della propria natura, in numerosi e differenti modi. Inoltre il comando OUT non implica alcuna memorizzazione di dati nella memoria.
Occorre perciò conoscere molto bene, oltre alla porta connessa con il dispositivo, anche l'esatto funzionamento del dispositivo stesso.

Per esempio, vi sono alcuni indirizzi di input/output i quali corrispondono a porte di input/output che possono soltanto ricevere dati (per esempio l'altoparlante): in questo caso ha senso utilizzare soltanto OUT.

Altri indirizzi corrispondono a porte che possono solo fornire dati (per esempio la tastiera): in questo secondo caso ha invece senso utilizzare unicamente IN. Altri indirizzi ancora possono sia ricevere che fornire dati (per esempio la porta di connessione del registratore): qui sarà allora possibile adoperare sia IN che OUT.
I dispositivi di input/output che di serie sono presenti sullo Spectrum sono l'altoparlante, l'interfaccia per il registratore a cassette e la tastiera.
Senza addentrarci troppo nel discorso, qui di seguito troverai due brevissimi esempi. Il primo modificherà, in base alla risposta, il colore del bordo dello schermo:

```
10 INPUT "SCRIVI UN NUMERO COMPRESO TRA 0 E 7 "; A
20 OUT 254, A
30 GOTO 10
```

Il secondo produrrà invece un'uscita sonora dall'altoparlante:

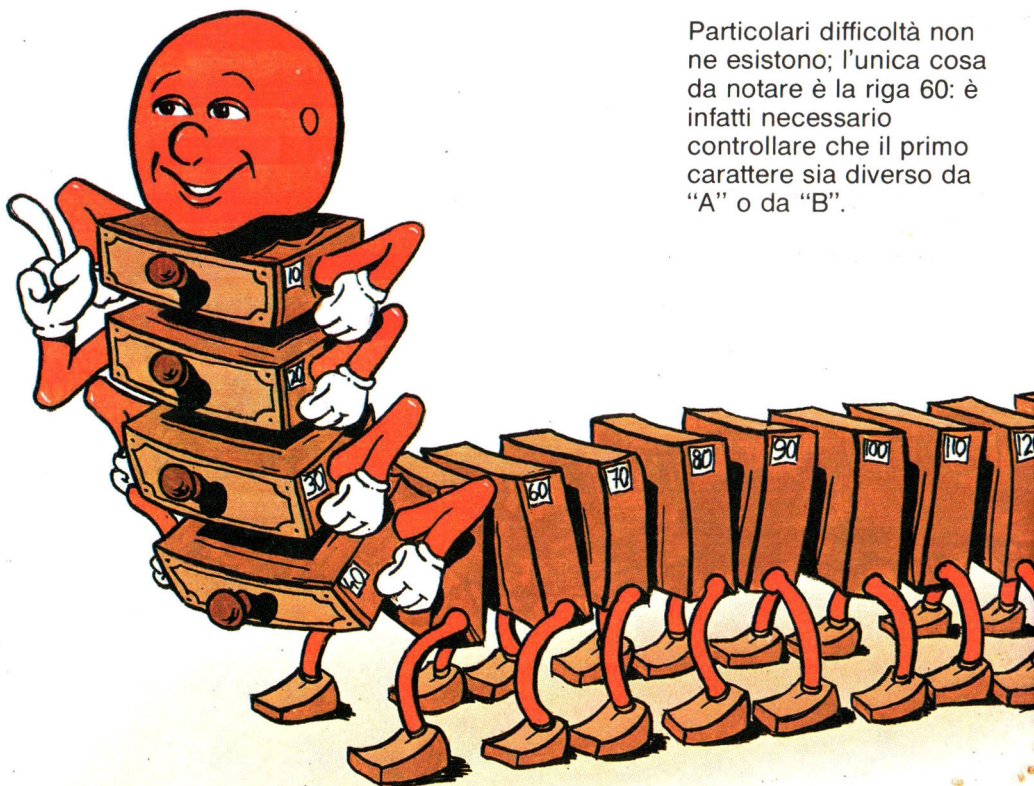
```
10 OUT 254, 16
20 OUT 254, 0
30 GOTO 10
```

Operazioni sulle stringhe

I programmi di oggi ti illustreranno alcune possibili applicazioni sulle funzioni per il trattamento delle stringhe.

Vediamo il primo esempio: stampare una stringa, introdotta dalla tastiera, solo se il primo carattere è diverso da "A" o da "B".

```
 10 PRINT "INSERISCI LA STRINGA"
 20 INPUT N$
 30 REM A$ = PRIMA LETTERA DELLA STRINGA
 40 LET A$ = N$ (1)
 50 REM CONTROLLA SE INIZIA PER A O B
 60 IF (A$ = "A") OR (A$ = "B") THEN GOTO 100
 70 REM LA STRINGA INIZIA CON UNA LETTERA DIVERSA
 80 PRINT "LA STRINGA È : "; N$
 90 GOTO 110
100 PRINT "LA STRINGA NON È STAMPABILE PERCHÉ INIZIA PER:" ; A$
110 STOP
```

Particolari difficoltà non ne esistono; l'unica cosa da notare è la riga 60: è infatti necessario controllare che il primo carattere sia diverso da "A" o da "B".

Secondo esempio: scrivere una parola - come al solito introdotta dalla tastiera - in senso inverso, cioè opposto al normale (da destra verso sinistra).

```
10 PRINT "INSERISCI LA STRINGA"
20 INPUT A$
30 LET K = LEN (A$)
40 FOR I = K TO 1 STEP - 1
50 PRINT A$ (I);
60 NEXT I
70 STOP
```

Anche in questo caso il procedimento è molto semplice: si preleva un carattere alla volta dalla parola (o dalla frase) da scrivere, partendo dalla destra e per tante volte quanti sono complessivamente i caratteri costituenti la parola stessa. Alla fine sullo schermo si troverà visualizzata la stringa originaria scritta al rovescio.
Se qualcosa non ti è chiaro, aggiungendo la riga

```
35 FOR J = 0 TO 500 : NEXT J
```

introducendo cioè un ciclo di ritardo, avrai la possibilità di osservare molto più agevolmente la formazione, carattere dopo carattere, della parola completa.
Un'altra possibile soluzione avrebbe potuto essere:

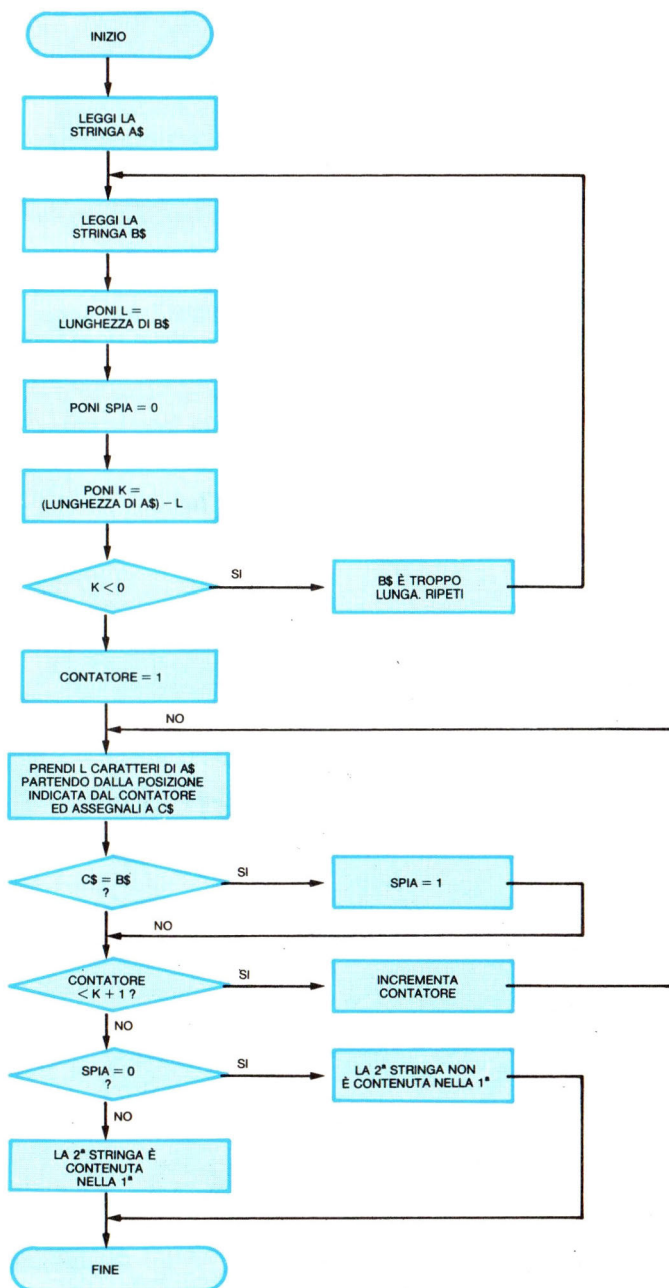
```
10 PRINT "INSERISCI LA STRINGA"
20 INPUT A$
30 LET B$ = " "
40 LET K = LEN (A$)
50 FOR I = 1 TO K
60 LET A$ = A$ (1 TO K - I + 1)
70 LET C$ = A$ (LEN (A$))
80 LET B$ = B$ + C$
90 NEXT I
100 PRINT B$
110 STOP
```

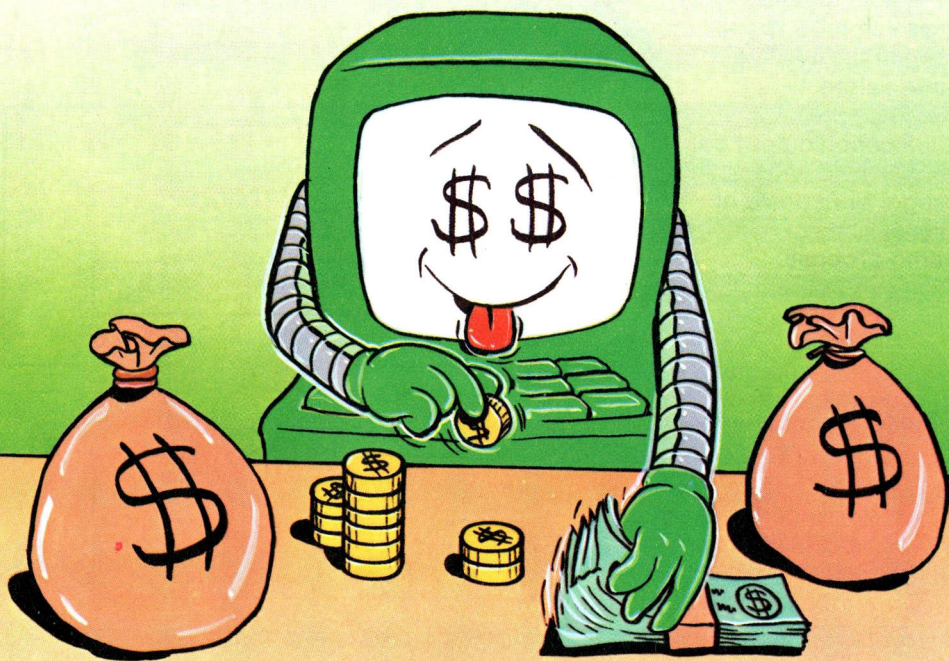
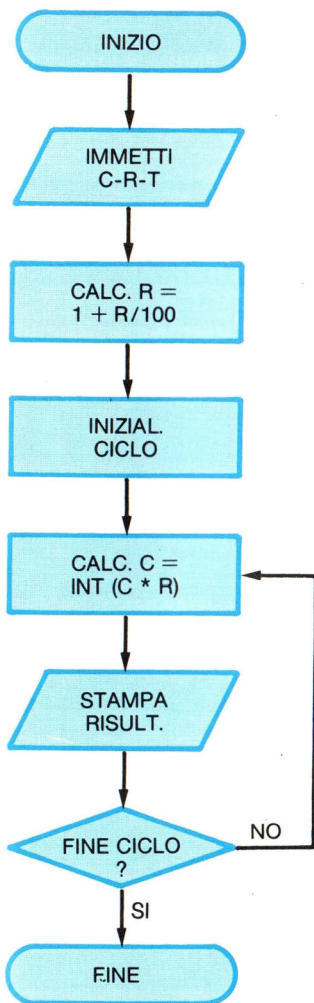
In questo caso i vari caratteri, anziché essere direttamente inviati sullo schermo, vanno ad aggiungersi, uno dopo l'altro, nella variabile B$. Terminato il ciclo (linee 30 - 70) B$ conterrà la parola invertita e la si potrà quindi stampare.

Come ultimo esempio vogliamo risolvere questo problema: cercare se una stringa è contenuta in un'altra stringa (esempio: la parola "carica" è contenuta in "scaricatori", mentre "mela" non ha nulla a che vedere con "pagina").
Vediamone innanzitutto una possibile risoluzione:

```
 10 INPUT "INSERISCI LA PRIMA STRINGA" 'A$
 20 INPUT "INSERISCI LA SECONDA STRINGA" 'B$
 30 LET L = LEN (B$)
 40 LET SPIA = 0
 50 LET K = LEN (A$) − L
 60 IF K < 0 THEN PRINT "TROPPO LUNGA! RIPETI." : GO TO 20
 70 FOR I = 1 TO K + 1
 80 LET C$ = A$ (I TO I + L − 1)
 90 IF B$ = C$ THEN LET SPIA = 1
100 NEXT I
110 IF SPIA = 0 THEN PRINT "LA SECONDA STRINGA NON È
    CONTENUTA NELLA PRIMA."
120 IF SPIA = 1 THEN PRINT "LA SECONDA STRINGA È
    CONTENUTA NELLA PRIMA."
130 STOP
```

Ed ecco alcuni commenti esplicativi.
Righe 10 - 20: vengono richieste in ingresso la due stringhe, assegnandone i valori alle variabili A$ e B$.
Righe 30 - 40: la variabile numerica L assume il valore dato dal numero di caratteri

componenti la variabile B$. SPIA è invece una variabile che potrà assumere due soli valori, 0 e 1; varrà 0, se A$ non contiene B$, 1, nel caso contrario.
Righe 50 - 60: se la lunghezza di B$ risulta superiore a quella di A$, è chiaro che il confronto non può essere possibile. Occorre ripetere l'inserimento di B$.
Righe 70 - 100: questo è il cuore del programma. Utilizzando la funzione MID$ si confrontano man mano i caratteri di A$ con quelli di B$. Nel caso in cui l'esito sia positivo (cioè B$ sia contenuto in A$), FLAG assume valore 1.
Righe 110 - 120: costituiscono l'output del programma, sono cioè i messaggi di risposta al problema che ci eravamo proposti.

Capitali e interessi

Il prossimo listato riguarda un programma di tipo finanziario: il calcolo dell'interesse composto e la stampa della relativa tabella ordinata per anno.
L'algoritmo utilizzato è molto semplice: si tratta, in sostanza di:
A) Richiedere i dati necessari, vale a dire:
1) Il capitale (C) su cui eseguire il calcolo.
2) Il tasso di interesse (R) (associa il nome della variabile al termine < RAGIONE >)
3) Il tempo (T).
B) Calcolare il montante (capitale + interesse) per ogni singolo anno, avendo cura di considerare come capitale per l'anno successivo il montante del precedente.
C) Stampare il prospetto dei vari anni, avendo cura che gli importi risultino bene incolonnati.

```
10 INPUT "C "; C' "R "; R' "T "; T
20 LET R = 1 + R/100
30 FOR A = 1 TO T
40 LET C = INT (C * R)
50 PRINT A; TAB 31 – LEN STR$ C; C
60 NEXT A
```

Per quest'ultima esigenza, sarà molto comodo utilizzare le funzioni stringa < STR $ > e < LEN > unitamente a < TAB >.
Poiché nei tuoi futuri programmi incontrerai molto spesso la necessità di un incolonnamento rigoroso dei dati, è bene che familiarizzi ora con questo tipo di istruzioni.

Commento al listato

10 INPUT dei valori di CAPITALE, TASSO (R) e TEMPO (N. ANNI)
20 Trasformazione del tasso di interesse da percentuale a decimale
30 Impostazione del ciclo in funzione degli anni
40 Calcolo del montante annuo
50 Stampa della tabella con N. anno e valore montante.
Nota in particolare l'istruzione TAB (31 – LEN STR$ C che ha il compito di posizionare correttamente il dato da stampare.
60 Chiusura del ciclo.

VIDEOESERCIZI

Annota nello spazio apposito il risultato da te previsto per ciascun esercizio proposto e poi verificalo con la soluzione del tuo computer. Se avrai commesso anche un solo errore, ripassa la lezione.

```
10 LET A$ = "12345"
20 LET B$ = "67"
30 LET C$ = "890"
40 PRINT LEN B$
50 PRINT LEN (A$ + B$ + C$)
60 PRINT LEN C$ – LEN B$
```

Per evidenziare

```
10 LET T$ = "TITOLO"
20 FOR C = 1 TO LEN T$
30 PRINT T$ (C); " ";
40 NEXT C
```

Prima di far girare il programma scrivine l'output

Indovina dove stampa le stringhe

```
10 CLS
20 INPUT "STRINGA ="; S$
30 FOR I = 1 TO 10 : PRINT : NEXT I
40 PRINT TAB (31 – LEN S$)/2; S$
50 PRINT AT 18, 0; "ANCORA?"
60 LET A$ = INKEY$ : IF A$ = " " THEN GOTO 60
70 IF A$ = "s" OR A$ = "S" THEN RUN
80 STOP
```

In quale riga?

Dove rispetto alle colonne?